comsar

Release 0.0.3

Michael Blaß

Apr 23, 2021

CONTENTS

1 Installation **3**
1.1 Installation from PyPi 3
1.2 Installation from source 3

2 Basic usage **5**

3 Track sytem **7**
3.1 PitchTrack 7
3.2 RhythmTrack 7
3.3 TimbreTrack 7
3.4 FormTrack 7

4 Open source **9**

Index **11**

The Computational Music and Sound Archiving system provides high-level audio feature extraction facilities for multi-viewpoint music similarity analysis.

Music similarity is hard to analyse. A viewpoint highlights certain aspects of musical perception. Asking for similarity regarding pitch requires another viewpoint than asking for rhythm similarity.

COMSAR combines pre-selected low-level audio features to *Track* objects, which represent a viewpoint.

CHAPTER

ONE

INSTALLATION

1.1 Installation from PyPi

comsar is available on PyPi. Simply run the following command in your favorite terminal emulator:

```
pip install comsar
```

1.2 Installation from source

Installation from source is done in two steps:

- If you do not have git installed, simply navigate to the source code repository, click on the green "Code" button and then select "Download ZIP". Otherwise, clone the source code with git:

  ```
  git clone https://github.com/ifsm/comsar
  ```

- Once the code is downloaded, change to the comsar root directory and advise Python to install the pacakge:

  ```
  cd path/to/comsar
  python3 -m pip install .
  ```

CHAPTER

TWO

BASIC USAGE

In order to compute audio features regarding a certain track, you just have to create an instance of your desired track object and then call its `extract()` method with the path to an audio file. Considre the following example:

```
from comsar.tracks import TimbreTrack

tt = TimbreTrack()
res = tt.extract('path/to/my_audio.wav')
res.to_pickle('my_features.pkl')
```

The first line imports the desired Track object, in this case a `TimbreTrack`. The third line creates a `TimbreTrack` instance with the name `tt`. The fourth line calls the `extract` method of `tt` and passes it the path to an actual audio file. comsar then processes the audio file and makes the results available under the name `res`. The fifths line eventually saves the results to disc.

CHAPTER

THREE

TRACK SYTEM

3.1 PitchTrack

3.2 RhythmTrack

3.3 TimbreTrack

class comsar.tracks.**TimbreTrack**(*stft_params: Optional[apollon.signal.container.StftParams] = None, corr_dim_params: Optional[apollon.signal.container.CorrDimParams] = None*)

High-level interface for timbre feature extraction.

__init__(*stft_params: Optional[apollon.signal.container.StftParams] = None, corr_dim_params: Optional[apollon.signal.container.CorrDimParams] = None*) → None

Parameters

- **stft_params** – Parameter for STFT.
- **corr_dim_params** – Parameter set for correlation dimension.

extract(*path*) → pandas.core.frame.DataFrame

Run TimbreTrack on audio file.

Parameters **path** – Path to audio file.

Returns Extracted features.

property n_features

Number of features.

Returns Number of audio features.

3.4 FormTrack

Implementation of the FormTrack is planed.

CHAPTER

FOUR

OPEN SOURCE

comsar is an open source project. It is published under the permissive BSD 3-Clause License. You may change and republish the code for any personal or commercial project. The comsar source code is available on GitHub.

INDEX

Symbols

`__init__()` (*comsar.tracks.TimbreTrack method*), 7

E

`extract()` (*comsar.tracks.TimbreTrack method*), 7

N

`n_features()` (*comsar.tracks.TimbreTrack property*), 7

T

`TimbreTrack` (*class in comsar.tracks*), 7